Lab Exercise: Parameter Passing

About Intertech

Thank you for choosing Intertech for your training from Udemy. The next page of this document will get you started with your lab if you'd like to skip ahead. Below is a brief overview of Intertech as well as a promo code for you for future live Intertech trainings.

Intertech (www.intertech.com) is the largest combined software developer **training** and **consulting** firm in Minnesota. Our unique blend of training, consulting, and mentoring has empowered technology teams in small businesses, Fortune 500 corporations and government agencies since 1991.

Our training organization offers live in-classroom and online deliveries, private on-site deliveries, and on-demand options such as the course in which you've enrolled from Udemy. We cover a broad spectrum of .NET, Java, Agile/Scrum, Web Development, and Mobile technologies. See more information on our training and search for courses by **clicking here**.

As a Udemy customer, you can save on any live in-classroom, live online, or onsite training with Intertech as well. Just use promo code "Udemy_Labs" when enrolling **and** save 35% on the course price.

We appreciate you choosing Intertech on Udemy for this course!

Lab Exercise

Parameter Passing

Objects, primitives, and String objects behave differently when passed to methods. In this experimentation lab, you explore how each behaves when passed to a method that uses and modifies its data.

Specifically, in this lab you will:

- Pass a MyDate object, integer primitive, and String object to separate modifying methods to see how they are affected
- See some additional String methods and operators
- Optionally explore the StringBuffer class and how it behaves when passed to a method

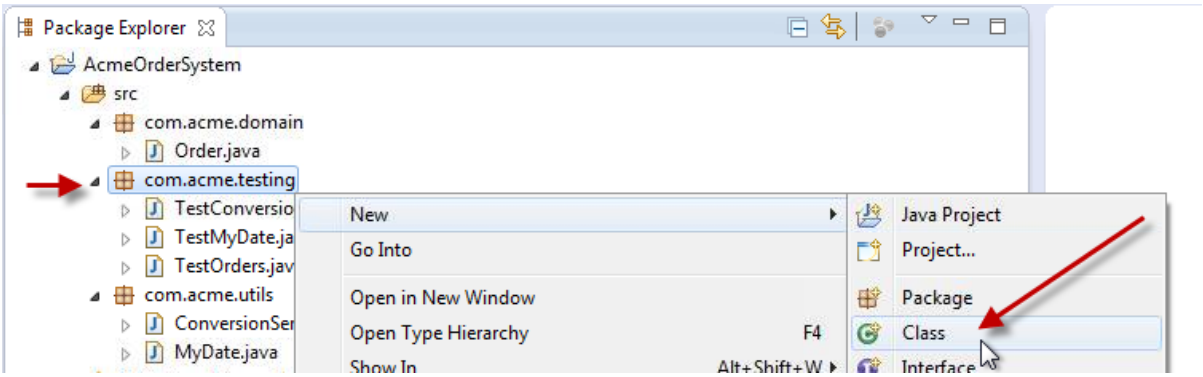
Scenario

This lab deviates for a bit away from the Acme Order System. The purpose of this lab is to allow you to see how objects and primitives are affected when passed as parameters to methods.

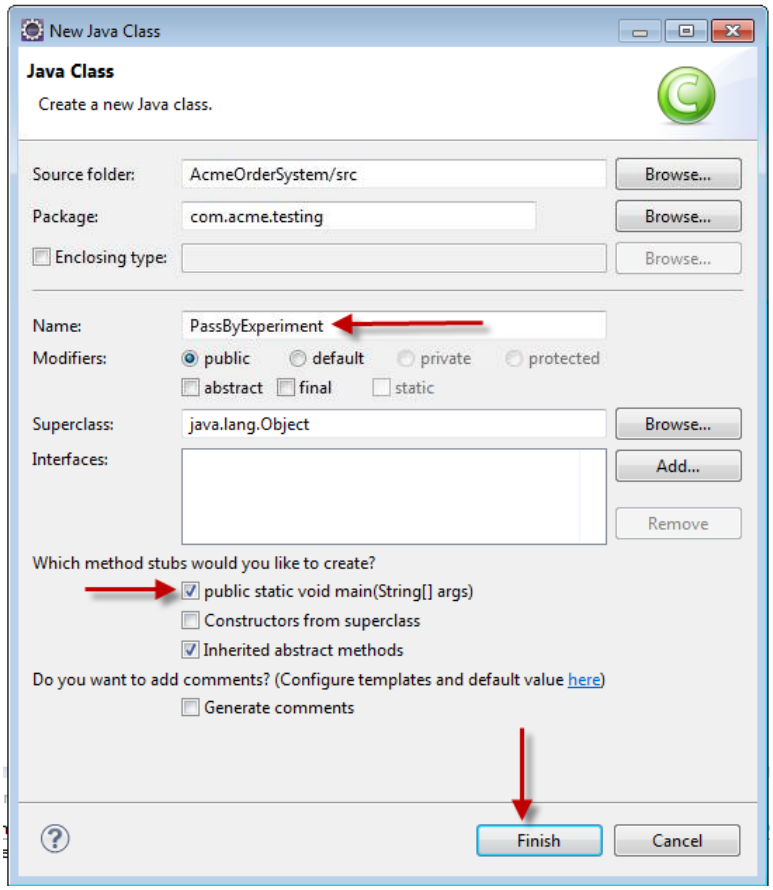
Step 1: Create a PassByExperiment class

In this step you create another testing class called PassByExperiment in the com.acme.testing package.

1.1 Create the PassByExperiment class. In the Package Explorer view, right-click on the com.acme.testing package in the AcmeOrderSystem project and select ***New > Class***.



1.2 In the New Java Class window, make sure the public static void main( ) method stub is checked in order to create the method. Enter PassByExperiment as the name of the class, and click the ***Finish*** button.



1.3 Add a static method to PassByExperiment called passObject( ) that gets passed a MyDate object and changes the year of the MyDate object.

```
public static void passObject(MyDate d){
  d.year=2009;
}
```

1.4 Add a static method to PassByExperiment called passPrimitive( ) that gets passed an int (the year of a MyDate object) and changes the int to some other year.

```
public static void passPrimitive(int i){
  i=2010;
}
```

1.5 Add a static method to PassByExperiment called passString( ) that gets passed a String that represents a month/day/year and replaces the year portion of the string with a new year. For example, if "1/20/2001" is passed in, "1/20/2012" is passed out.

```
public static void passString(String s){
  int yearSlash =s.lastIndexOf('/');
  s = s.substring(0, yearSlash+1);
  s += "2012";
  System.out.println("New date string: " + s);
}
```

Notice your use of a couple of the String methods (java.lang.String) and operators (+ for concatenation and += for assignment) to get this to work.

1.6 Save your file and fix any compiler errors before moving to the next step.

Note: As this new testing class is in a different package than the MyDate class that is getting used, what do you think is required to allow PassByExperiment to use the MyDate class? That's right — an import statement needs to go at the top of the page.

```
import com.acme.utils.MyDate;
```

Note: There is a shortcut inside Eclipse to help with imports. You can press ***Control-Shift-O (windows) or Command-Shift-O (Mac)*** to have Eclipse automatically add the appropriate import statements at the top of a class. If Eclipse cannot tell which class is being used (because two classes may have the same name), you'll be prompted to indicate which import to use.

Step 2: Test the pass methods

2.1 Add code to the main( ) method to test each of the "pass" methods.

2.1.1 In the main( ) method, first create a MyDate object.

```
MyDate date = new MyDate(1,20,2008);
```

2.1.2 Print out the date reference before and after calling the passObject( ) method.

```
System.out.println("Before passing an object " + date);
passObject(date);
System.out.println("After passing an object " + date);
```

2.1.3 Do the same work to write out the date's year and call on the passPrimitive( ) method.

```
System.out.println("Before passing a primitive  " + date.year);
passPrimitive(date.year);
System.out.println("After passing a primitive  " + date.year);
```

2.1.4 Lastly, create a String reference from toString( ) of the MyDate object. Use this String object to call on the passString( ) method. Once again print out the String value before and after the call to the method.

```
String x = date.toString();
System.out.println("Before passing a String  " + x);
passString(x);
System.out.println("After passing a String  " + x);
```

2.2 Run the PassByExperiment class.

2.2.1 Before executing the code, how do you expect each method to affect the objects or primitive being passed? Document your hypothesis on what you expect to be displayed by filling in the blanks below.

Before passing an object 1/20/2008

After passing an object _____

Before passing a primitive 2009

After passing a primitive _____

Before passing a String 1/20/2009

New date string: 1/20/2010

After passing a String _____

2.2.2 Right-click on PassByExperiment and select *Run As > Java Application*.

2.2.3 When you run the test, the output in the Console view should look like the following. Did this match your expectations?

```
Before passing an object 1/20/2008
After passing an object 1/20/2009
Before passing a primitive  2009
After passing a primitive  2009
Before passing a String  1/20/2009
New date string: 1/20/2012
After passing a String  1/20/2009
```

Lab Solution

PassByExperiment.java

```
package com.acme.testing;

import com.acme.utils.MyDate;

public class PassByExperiment {

  public static void main(String[] args) {
    MyDate date = new MyDate(1, 20, 2008);
    System.out.println("Before passing an object " + date);
    passObject(date);
    System.out.println("After passing an object " + date);

    System.out.println("Before passing a primitive  " +
date.year);
    passPrimitive(date.year);
    System.out.println("After passing a primitive  " +
date.year);

    String x = date.toString();
    System.out.println("Before passing a String  " + x);
    passString(x);
    System.out.println("After passing a String  " + x);
  }

  public static void passObject(MyDate d) {
    d.year = 2009;
  }

  public static void passPrimitive(int i) {
    i = 2010;
  }

  public static void passString(String s) {
    int yearSlash = s.lastIndexOf('/');
    s = s.substring(0, yearSlash + 1);
    s += "2012";
    System.out.println("New date string: " + s);
  }
}
```

Bonus Lab

Step 3: Replace String with StringBuilder

The use of String objects, although convenient, may not always be best suited in cases where a lot of String manipulation occurs. Explore the Java documentation and look up the StringBuilder class in the java.lang package.

3.1 Add a passStringBuilder( ) method. Create a new method that does the same work that passString( ) does, but works with a StringBuilder object rather than a String object.

3.2 In the main( ) method, create a StringBuilder object reference, and pass this to the new passStringBuilder( ) method. Print out the StringBuilder before and after the call.

3.3 Run the test again. How is a String object different from a StringBuilder with regard to being passed to a method? Does a StringBuilder reference behave more like a String, a primitive, or an object when passed to a method?

Give Intertech a call at **1.800.866.9884** or visit Intertech's website.